Decidability and Computability

With an informal proof of The Halting Problem

Zhewen Zheng

December 6, 2018

University of Washington

Decidability

Decidability

What does it mean that something is DECIDABLE?

A true/false problem is decidable if there exists an **effective method** for deriving the correct answer.

An effective method is formally defined as a mechanical process satisfying the following criteria:

1. It consists of a finite number of exact, finite instructions.
2. When it is applied to a problem from its class:
    2.1 It always finishes after a finite number of steps.
    2.2 It always produces a correct answer.
3. In principle, it can be done by a human without any aids except writing materials.
4. Its instructions need only to be followed **rigorously** to so succeed.

Decidability Cont.

We have learned Propositional Logic(a.k.a Sentential Logic) and Predicate Logic.

- What is the difference between these two?

Decidability Cont.

We have learned Propositional Logic(a.k.a Sentential Logic) and Predicate Logic.

- What is the difference between these two?
- Is Propositional Logic decidable? Why or Why not?

Decidability Cont.

We have learned Propositional Logic(a.k.a Sentential Logic) and Predicate Logic.

- What is the difference between these two?
- Is Propositional Logic decidable? Why or Why not?
- Is Predicate Logic decidable? Why or Why not?

Computability

Computability

What does it mean that something is COMPUTABLE?

A mathematical problem or a function is computable if it can be solved in principle by a computing device. Computing device means something that carries the capability of computation, thus given a set of instruction and input the device returns and output.

Notation

Use $f(x)\downarrow$ to say the computation halt, or converges.

Use $f(x)\uparrow$ to say the computation fails to halt, or diverges.

$f(x)\downarrow = y$ can be used to say the computation halts with an output of y.

The Halting Problem

The Halting Problem

What is The halting problem?

The problem is to determine, given a program and an input to the program, whether the program will eventually halt when run with that input. In this abstract framework, there are no resource limitations on the amount of memory or time required for the program's execution; it can take arbitrarily long, and use an arbitrary amount of storage space, before halting. The question is simply whether the given program will ever halt on a particular input.

A simple proof

We first define a function that does exactly what the Halting Problem asks for:

$$f(x) = \begin{cases} 1, & \text{if } \varphi_e(e)\downarrow. \\ 0, & \text{if } \varphi_e(e)\uparrow. \end{cases} \tag{1}$$

Then we explore the computability of f by define g:

$$g(x) = \begin{cases} \varphi_e(e) + 1, & \text{if } f(e) = 1. \\ 0, & \text{if } f(e) = 0. \end{cases} \tag{2}$$

Thus we can easily see that g(x) is not computable. By definition of computability, we can see if f(x) is computable, then g(x) follows. Here g(x) is not computable, thus f(x) is also not computable.

References i

W. Rebecca.
Computability theory.
American Mathematical Soc,2012

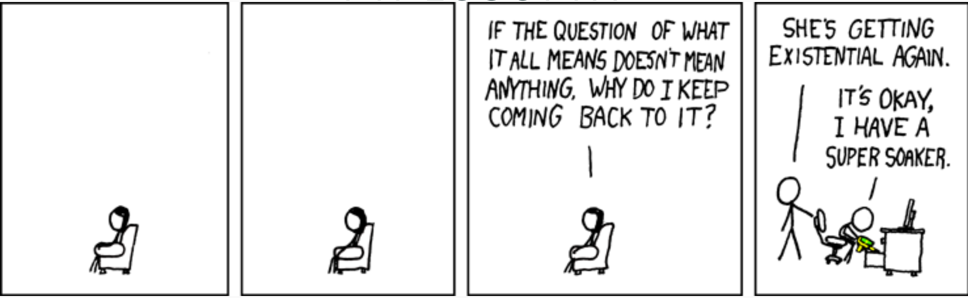
I. Neil.
Computability and Complexity.
The Stanford Encyclopedia of Philosophy, 2018

Wikipedia contributors
Halting Problem.
Wikipedia, avaliable at
https://en.wikipedia.org/wiki/Halting_problem, 2018

Wikipedia contributors
Decidability (logic)
Wikipedia, avaliable at
https://en.wikipedia.org/wiki/Decidability_(logic), 2018

References ii

Philosophy


IF THE QUESTION OF WHAT IT ALL MEANS DOESN'T MEAN ANYTHING, WHY DO I KEEP COMING BACK TO IT?
SHE'S GETTING EXISTENTIAL AGAIN.
IT'S OKAY, I HAVE A SUPER SOAKER.

https://xkcd.com/220/